Universidade Federal da Paraíba

Centro de Informática

Curso de Engenharia de Computação

Relatório - CI SmartLift

Ana Flávia Silva Aragão Moura

Marismar da Costa Silva

Rebeca Andrade Medeiros

João Pessoa, 2017

Centro​ ​de​ ​Informática

Universidade​ ​Federal​ ​da​ ​Paraíba

Curso de Engenharia de Computação

Relatório - CI SmartLift

Ana Flávia Silva Aragão Moura

Marismar da Costa Silva

Rebeca Andrade Medeiros

Relatório elaborado para o projeto final da disciplina Circuito Lógicos II, ministrada pelo Professor Eudisley Gomes Dos Anjos do Centro de Informática da Universidade Federal da Paraíba.

João Pessoa, 2017

**Resumo**

Este relatório apresenta os processos e resultados elaborados a partir da execução do projeto do elevador CI SmartLift. O trabalho foi desenvolvido com o kit DE2-115 da Altera e utilizando a linguagem VHDL Verilog. O objetivo foi elaborar o sistema de um elevador utilizando chaves, LEDs, displays de sete segmentos e o LCD. Os estados deste elevador foram modelados com uma máquina de estados, com três estados: parado, subindo e descendo. Para funcionar o usuário tem que selecionar uma chave de SW0 até SW8 e em seguida pressionar o botão para que a escolha do andar de destino seja validada. Enquanto este processo está acontecendo dois displays (HEX0 e HEX1) mostram respectivamente o andar selecionado e o andar pelo qual o elevador está passando. Ao mesmo tempo o LCD mostra o estado atual do elevador. E por fim os LEDs simulam o abrir e fechar das portas ao atingir o andar selecionado e nos momentos de subida e descida.

**Palavras-chave:** Verilog, Elevador, FPGA, Chaves, LCD, LEDs e Botão.

**Lista de siglas**

VHDL - Hardware Description Language (Linguagem de descrição de hardware)

FPGA - Field Programmable Gate Array (Arranjo de portas programáveis em campo)

IEEE - O Instituto de Engenheiros Eletricistas e Eletrônicos

IDE - Integrated Development Environment (Ambiente de desenvolvimento integrado)

**Sumário**

1. Introdução 6

2. Metodologia 7

3. Funcionamento

3.1. Descrição 10

3.2. Digramas e Esquemático 10

4. Execução

4.1. Resultados 13

4.2. Dificuldades 14

4.3. Sugestões 15

5. Conclusões 16

6. Referências 17

7. Apêndice 18

**1. Introdução**

Este relatório tem como objetivo expor os processos realizados e resultados obtidos durante a elaboração do projeto CI SmartLift, um elevador com oito andares e térreo, como atividade final da disciplina de Circuitos Lógicos II.

Para a realização desse projeto foi utilizado o conteúdo ensinado durante todo o período letivo, com a VHDL Verilog e a placa FPGA. Para esse projeto foi formado um grupo composto de três alunas do curso de Engenharia de Computação. O sistema foi idealizado e implementado seguindo as orientações estabelecidas de funcionamento, de que o elevador teria que possuir três estados: partida, parado, subindo e descendo.

Verilog foi desenvolvida no ano de 1985 pela empresa privada Gateway Design Automation, a empresa foi comprada e em 1990 a linguagem passou a ter domínio público, a padronização atual da linguagem é feita pelo IEEE. Possui semelhanças com outras linguagens de programação como C e C++. A placa FPGA foi criada por Ross Freeman em 1984, é um circuito integrado baseado em uma matriz de blocos lógicos e de conexões programáveis, uma de suas vantagens é que o usuário pode utilizar a placa de acordo com a sua necessidade.

Durante a execução do projeto foram utilizados conceitos e aplicações aprendidas durante as aulas, algumas delas: blocos always, LEDs, chaves, botões, displays de 7 segmentos, LCD. E também foi aplicada para a mudança de estados do elevador a máquina de estados.

O objetivo final deste trabalho é que o elevador seja capaz de ser movido por todos os andares de acordo com a escolha do usuário e que durante isso um display exibe o andar atual do elevador e o outro o andar selecionado. Também o LCD mostrasse o sentido do movimento ou a ausência deste. E que os LEDs simulem o abrir e fechar das portas.

**2. Metodologia**

Será abordado todas as etapas mediante no desenvolvimento do projeto. Tal que, a explicação detalhada de toda a ação desenvolvida no percurso do trabalho. No qual, usaremos diversos tipos de diagramas para um melhor entendimento e explicação do projeto, bem como, os instrumentos utilizados, dificuldade enfrentadas, tempo de execução e divisão de tarefas entre a equipe.

Presentes na plataforma de desenvolvimento, sendo utilizado, existem nove interruptores ou chaves, um botão de pressão, dois LEDs e o LCD. E necessário a criação de um módulo que realize o debouncing dos botões, devido ao ruído mecânico e de transição que estes introduzem nas mudanças de estado. Foi usado apenas um botão que será utilizado quando o usuário escolher o andar, que só irá mudar se ele apertar o botão. Existem ainda na placa oito LEDs, no qual, foram utilizados dois LEDs, um verde sinalizando a porta aberta do elevador e um vermelho sinalizando porta fechada.

O display de sete segmentos é um dispositivo que informa ou representa a composição de sete segmentos que podem acender ou apagar, de tal maneira que se determinados segmentos estiverem acessos, representa um número ou letra, por exemplo. Neste caso, usamos dois displays de sete segmentos no qual um mostrava o andar selecionado, enquanto o outro mostrava o andar atual.

A plataforma de desenvolvimento utilizada durante o projeto, integrada um LCD de duas linhas, cada uma com dezesseis caracteres visíveis e memória para um total de quarenta caracteres por linha. A interface entre a FPGA e o LCD é constituída por quatro linhas de dados, três linhas de controlo e duas de alimentação. As mensagens exibidas no LCD foram: subindo, descendo e parado. Junto os com caracteres representativos: \_, + e -. Além dos pinos setados dos elementos que vamos utilizar, vejamos as tabelas a seguir:

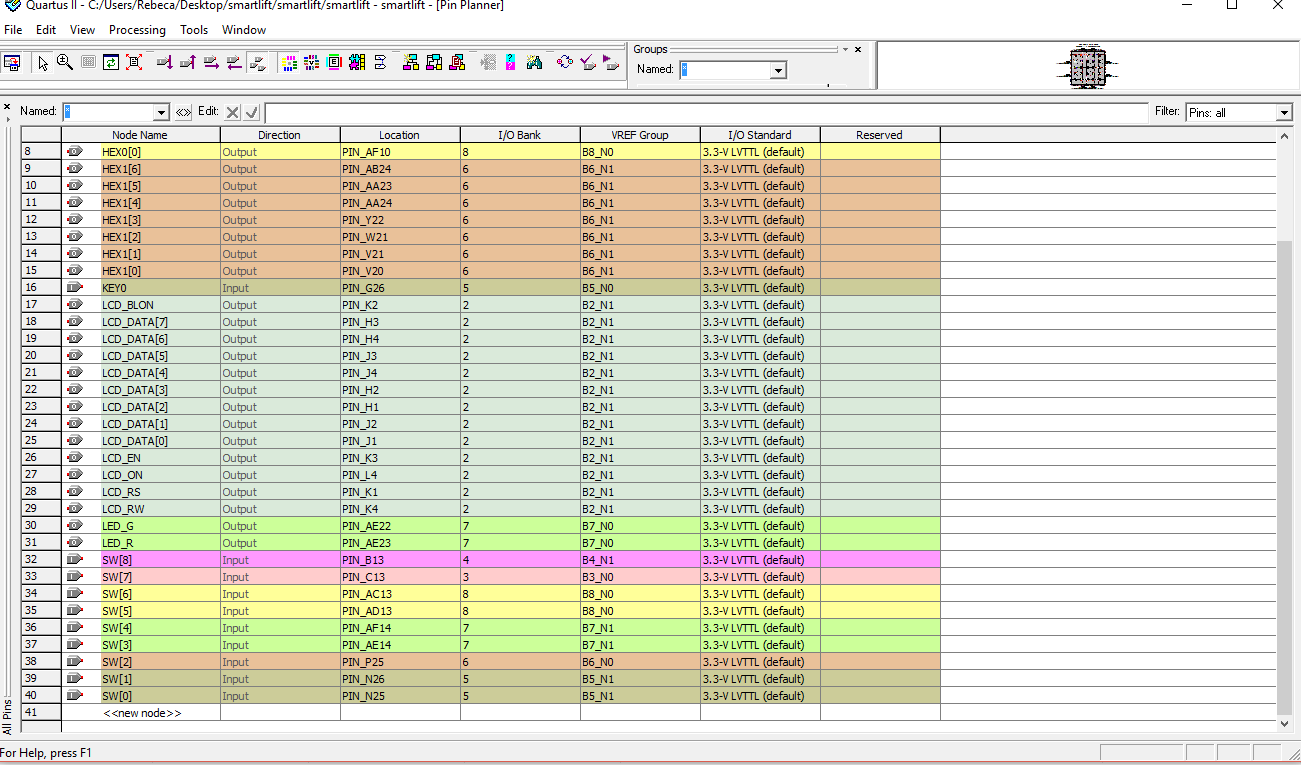


Figura 1 – Pinos.

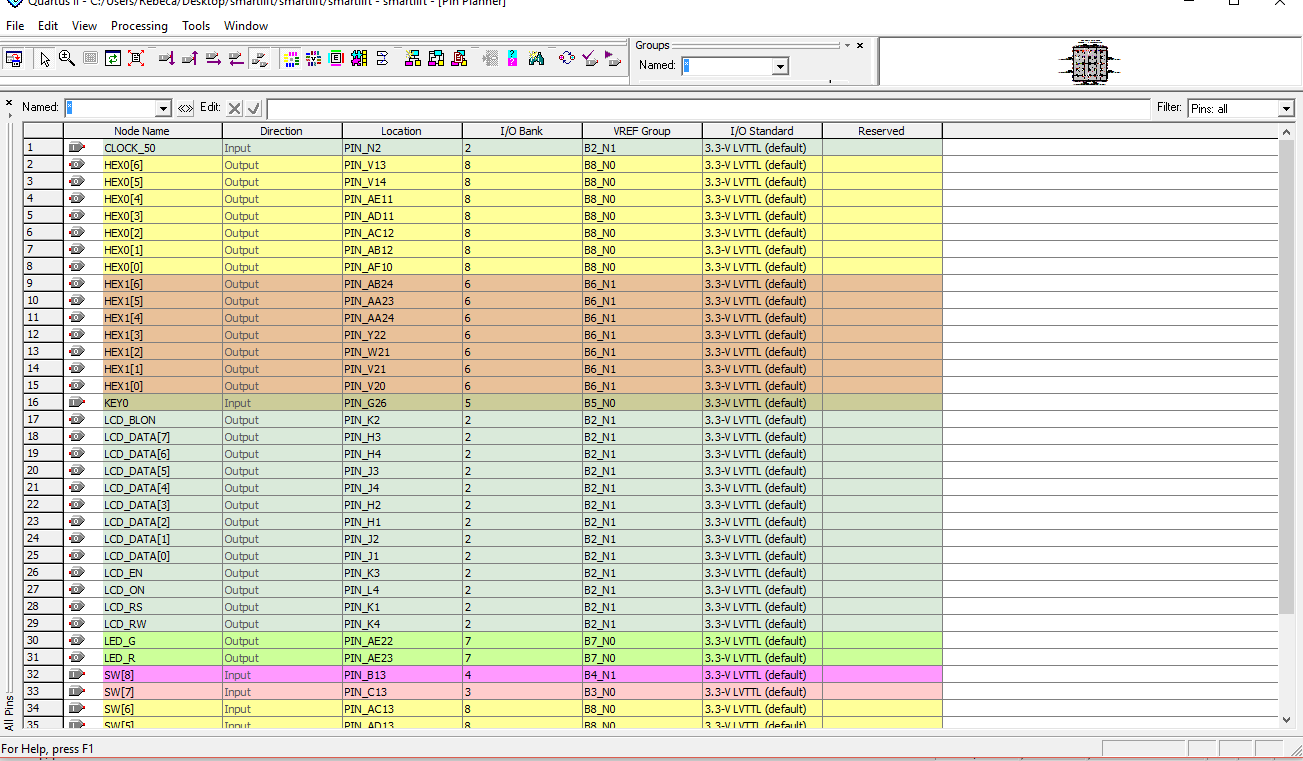


Figura 2 – Pinos.

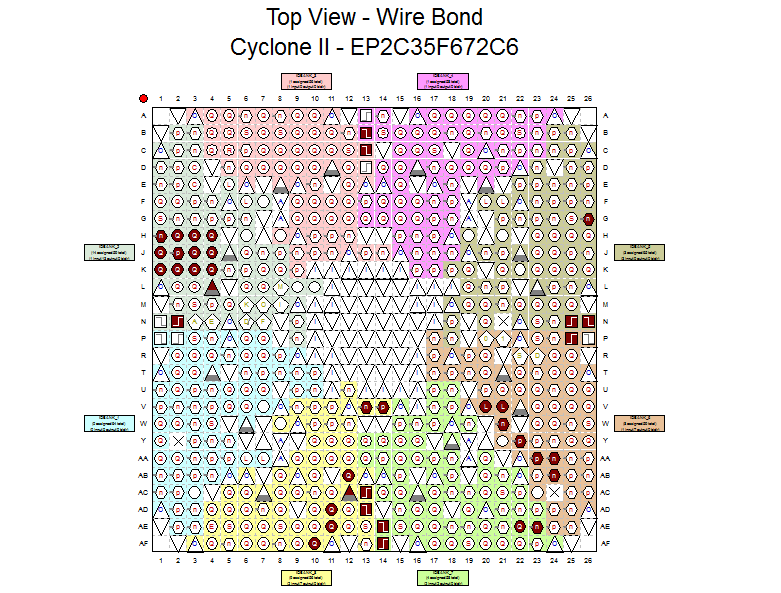


Figura 3 – Pinos.

Depois de setar os pinos, desabilitamos os pinos que não serão ser utilizado, no modo tri-stage.

Utilizando a linguagem de especificação de hardware VHDL realiza-se a conversão dos módulos para Máquinas de Estados Finitos que realizam o controle do processo do elevador. A Máquina de Estado Finito que realiza o controle do elevador , expresso sobre a linguagem de especificação VHDL. Na figura abaixo está presente a parte inicial do código.

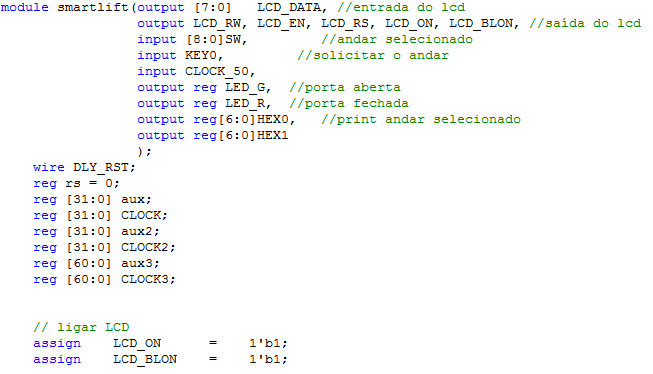


Figura 4 –Parte do código em Verilog.

O tempo de execução do projeto foram cerca de cinco dias, para a implementação do projeto. Sendo prescindível a contribuição dos monitores, bem como do professor da disciplina. Para um melhor entendimento das funcionalidades , da criação da maquina de estado e de otimização do código em VHDL.

**4. Funcionamento**

**4.1. Descrição**

O principal objetivo deste projeto é o funcionamento do elevador CI SmartLift, utilizando o kit de desenvolvimento Altera DE2-115, com o Cyclone II, para a execução dos estados foi utilizada uma máquina de estados finita. Esse elevador possui oito andares e um térreo (com chaves de SW0 até SW8), que são selecionados pelas chaves da placa FPGA e validados pelo botão. Todas as vezes que um andar é escolhido e o botão acionado, o display HEX0 passa a mostrar esse andar e o HEX1 os andares pelos quais o elevador está passando.

Ao todo o elevador possui três estados: parado, subindo e descendo. O estado vai ser parado todas as vezes que andar selecionado se igualar ao antar atual, esse estado mudara apenas quando um novo andar for escolhido. O estado subindo vai acontecer quando o selecionado for maior que o atual, então o elevador vai subir até atingir o andar escolhido e esperar por um novo comando. O estado descendo acontece apenas quando o andar selecionado for menor que o andar atual, então o elevador vai fazer um movimento de descida até atingir o andar. Todos esses estados vão ser mostrados durante a execução do projeto no LCD. Os LEDs verde e vermelho, representam o abrir e fechar das portas.

**4.2. Diagramas e Esquemático**

Vamos entender melhor o projeto ao analisarmos o diagrama de estados e a tabela de transição.

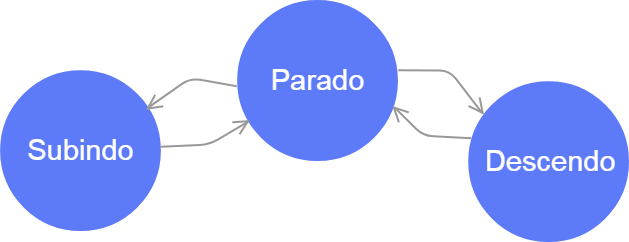
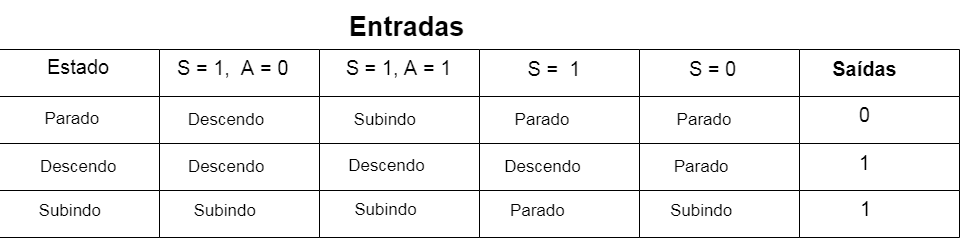


Figura 5 - Diagrama de Estados.

Tabela 1 - Tabela de transição de estados.



A identificação de validação para um algoritmo e parâmetros de entradas e saída, construção de processos, indexação, variável de controle e estados, resumindo em um fluxograma.

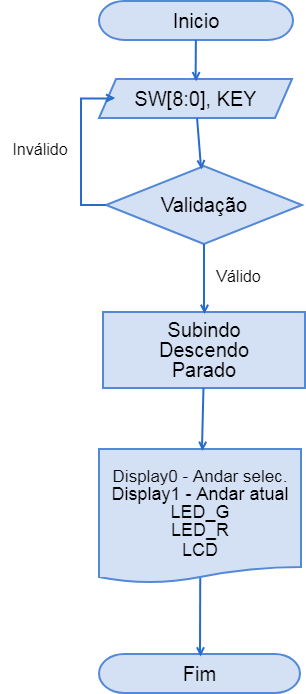


Figura 6 – Fluxograma.

Inicialmente os dados de entrada são coletados e registrados, dependendo do andar selecionado as informações são processadas. Sendo assim as saídas os LEDs, LCD e displays de 7 segmentos.

No esquemático podemos observar a organização dos parâmetros de entrada saídas, por exemplo.

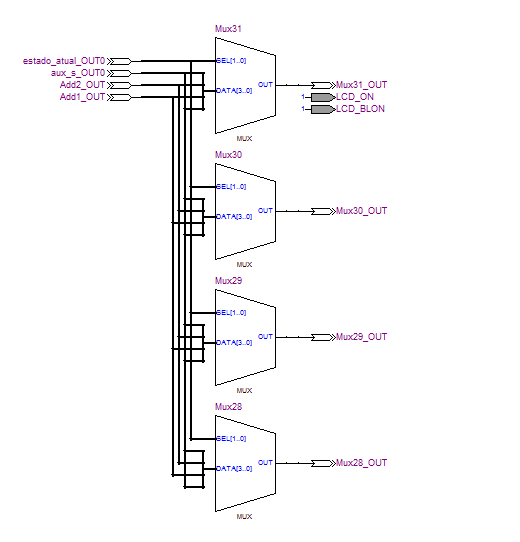
****

Figura 7 – Esquemático.

**4. Execução**

**4.1. Resultados**

Os resultados apresentados a seguir foram obtidos durante a execução do kit de desenvolvimento DE2-115 e o código encontrado no apêndice deste relatório. A figura abaixo mostra o estado do elevador após a chave de seleção do terceiro andar ter sido escolhida pelo usuário e o botão apertado, o display da direita mostra o andar escolhido e o da esquerda mostra o andar pelo qual o elevador está passando no momento. Como o andar selecionado foi maior que zero o elevador começou um movimento de subida, que está sendo mostrado no LCD. E o LED vermelho está aceso pois está reconhecendo que o estado atual do elevador não é mais parado, então as portas não podem estar abertas.

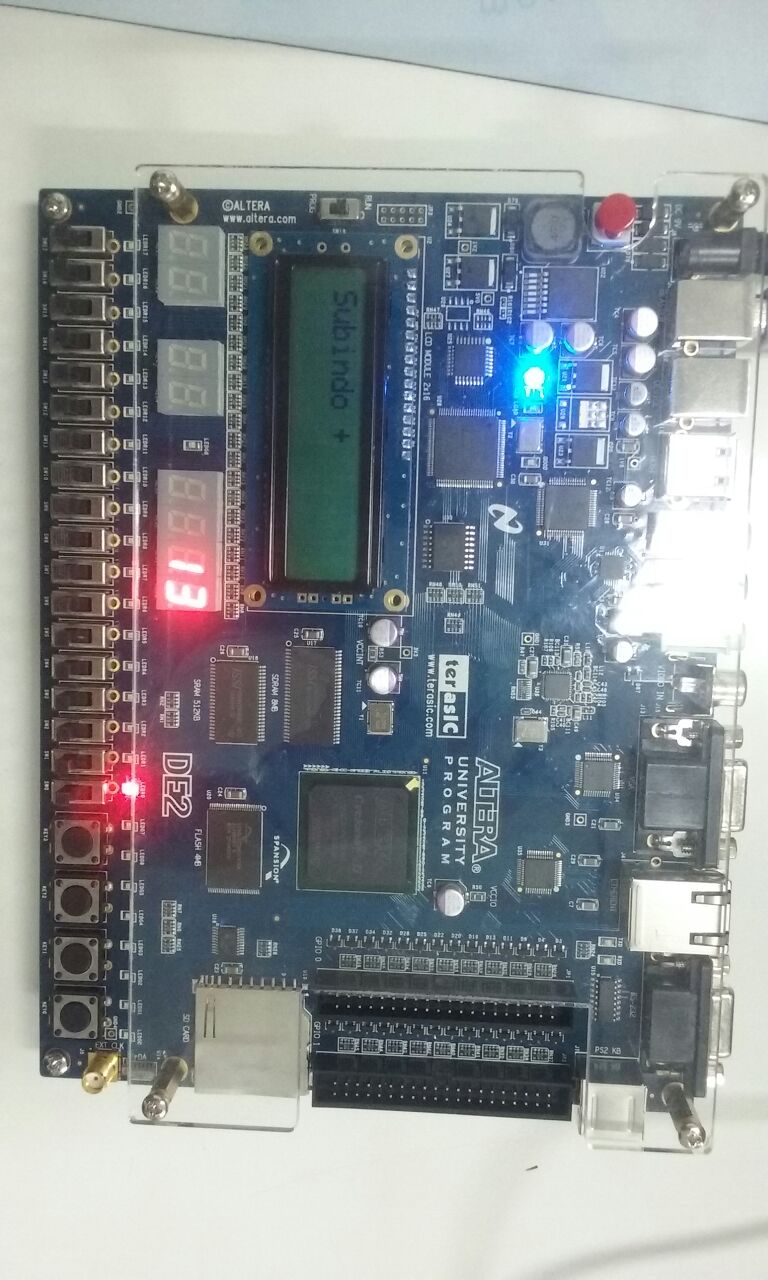


Figura – Movimento de subida do elevador.

Na figura a seguir o elevador atingiu um novo estado e agora está parado, pois o andar selecionado e o pelo que está passando possuem agora o mesmo valor. Assim o LCD agora mostra o estado atual como parado e o LED verde está aceso, pois as portas do elevador estão abertas.

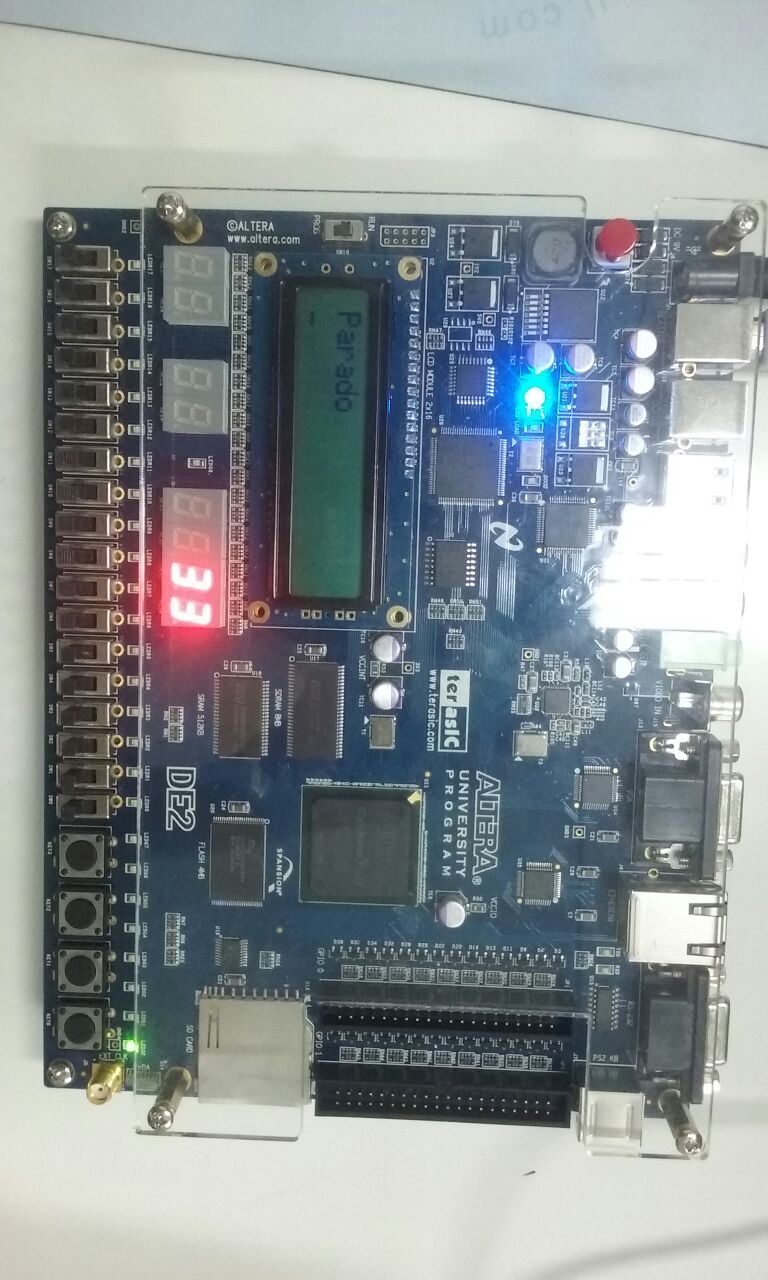


Figura – Movimento de parada do elevador.

Analisando os resultados apresentados podemos perceber que o elevador funcionou de forma desejada e apresentou o resultado esperado. Primeiro, as mudanças de andares aconteceram apenas quando a chave era selecionada e o botão apertado, os displays se comportaram de forma correta mostrando no display da esquerda os andares pelos quais o elevador estava passando e no da direita o andar escolhido. O LCD apresentou a mudança de estados (subindo e descendo, parado) quando esperado e os LEDs acenderam de acordo com os estados, parado com LED verde e subindo ou descendo com LED vermelho.

**4.2. Dificuldades**

Uma das dificuldades na elaboração deste projeto foi a implementação do LCD para que este mostrasse a mudança de estados pela qual o elevador estava passando. Nos primeiros testes apenas mostrava o seguinte estado: Parado \_, mesmo com mudanças na movimentação o estado continuava o mesmo.

Para que isso fosse contornado foi necessária uma mudança na máquina de estados do projeto. Inicialmente seria composta por oito andares e o andar zero, mas como os resultados desejados do LCD não foram obtidos os estados passaram de nove para apenas três: parado, subindo e descendo.

Depois dessa mudança os resultados esperados foram mostrados.

Outra dificuldade foi a inclusão de um estado inativo. Esse estado deveria ser ativado quando o elevador estivesse parado com os dois displays mostrando o mesmo andar durante determinado tempo, quando isso acontecesse o andar atual seria decrementado até atingir zero e deveria ficar em zero até que uma chave e o botão fossem ativados. Mesmo com diversos testes e mudanças no código esse estado não mostrou resultados positivos, o que o fez ser retirado na finalização deste projeto.

**4.3. Sugestões futuras**

Algumas sugestões para projetos futuros são: a implementação do estado inativo, adição de um número maior de LEDs para simbolizar o abrir e fechar das portas, maior quantidade de andares.

A inclusão de um botão por dentro e outro por fora do elevador. Sinais sonoros para indicar a chegada do elevador ao andar desejado, funcionando ao mesmo tempo que os LEDs.

**5. Conclusões**

Após a finalização do projeto CI SmartLift, podemos concluir que os objetivos e orientações estabelecidos foram cumpridos. O elevador funcionou da forma esperada, mostrando resultados claros. Todos os dispositivos propostos para uso foram implementados, com a adição de um botão para validar a seleção das chaves.

Além disso como atividade final da disciplina reforçou e trouxe para uma esfera mais prática as aplicações da linguagem Verilog e do FPGA, proporcionando aos alunos uma maior capacitação ao uso destes dois e possibilidades para pesquisa e trabalho em futuros projetos de outras disciplinas.

**6. Referências**

MACHADO, R. J. S. (2010). *Biblioteca de módulos Verilog para interface de FPGAs com periféricos I/O*. Disponível em < https://repositorio-aberto.up.pt/bitstream/10216/65328/1/000149183.pdf >. Acessado em 30 de novembro de 2017.

BRAGA, R. J. G.; DAMBROS, T.; ZANETTI, V. A. (2010). *Projeto elevador (Equipamento para transporte de pessoas)*. Disponível em < http://embedded.microprocessadores.com.br/wp-content/uploads/2015/06/docs/user123\_Documentacao-Elevador.pdf >. Acessado em 27 de novembro de 2017.

*What is na FPGA?.* Disponível em < https://www.xilinx.com/products/silicon-devices/fpga/what-is-an-fpga.html>. Acessado em 27 de novembro de 2017.

CURZEL, J. F.; LUDERS, R. (2014). *Modelagem e implementação em FPGA do controle discreto de uma planta didática de separação de itens.* Disponível em < http://www.joinville.ifsc.edu.br/~jlcurzel/PROJETO%20INTEGRADOR/Artigos%20Prof.%20Jeferson/2014\_CBA.pdf >. Acessado em 30 de novembro de 2017.

LIMA, T. (2016). *Tutoria de Verilog: Conversor BCD para 7 segmentos.* Disponível em < https://www.embarcados.com.br/tutorial-de-verilog-conversor-bcd-para-7-segmentos/ >. Acessado em 30 de novembro de 2017.

**7. Apêndice**

**7.1. Código principal**

module smartlift(output [7:0] LCD\_DATA, //entrada do lcd

output LCD\_RW, LCD\_EN, LCD\_RS, LCD\_ON, LCD\_BLON, //saída do lcd

input [8:0]SW, //andar selecionado

input KEY0, //solicitar o andar

input CLOCK\_50,

output reg LED\_G, //porta aberta

output reg LED\_R, //porta fechada

output reg[6:0]HEX0, //print andar selecionado

output reg[6:0]HEX1

);

wire DLY\_RST;

reg rs = 0;

reg [31:0] aux; //auxiliar para o clock

reg [31:0] CLOCK;

reg [31:0] aux2;

reg [31:0] CLOCK2;

reg [60:0] aux3;

reg [60:0] CLOCK3;

// ligar LCD

assign LCD\_ON = 1'b1;

assign LCD\_BLON = 1'b1;

//seg - (a, b, c, d, e, f, g)

//seg - 1 desligado e 0 ligado

integer s; //andar solicitado

integer aux\_s; //andar atual

reg [1:0] estado\_atual;

parameter parado = 0, subindo = 1, descendo = 2; // parado subindo descendo

always @(posedge CLOCK\_50)begin

if(aux == 0)begin

aux <= 24999999;

CLOCK <= ~CLOCK;

end else begin

aux <= aux - 1;

end

end

always @(posedge CLOCK\_50)begin

if(aux2 == 0)begin

aux2 <= 49999999;

CLOCK2 <= ~CLOCK2;

end else begin

aux2 <= aux2 - 1;

end

end

always @( negedge KEY0 ) begin

case (SW)

9'b000000001: begin

HEX0 = 7'b1000000; //se o andar selecionado for 0 s = 0;

end

9'b000000010: begin

HEX0 = 7'b1111001; //se o andar selecionado for 1

s = 1;

end

9'b000000100: begin

HEX0 = 7'b0100100; //se o andar selecionado for 2

s = 2;

end

9'b000001000: begin

HEX0 = 7'b0110000; //se o andar selecionado for 3

s = 3;

end

9'b000010000: begin

HEX0 = 7'b0011001; //se o andar selecionado for 4

s = 4;

end

9'b000100000: begin

HEX0 = 7'b0010010; //se o andar selecionado for 5

s = 5;

end

9'b001000000: begin

HEX0 = 7'b0000010; //se o andar selecionado for 6

s = 6;

end

9'b010000000: begin

HEX0 = 7'b1111000; //se o andar selecionado for 7

s = 7;

end

9'b100000000: begin

HEX0 = 7'b0000000; //se o andar selecionado for 8

s = 8;

end

default: begin

HEX0 = 7'b1110111; //default: \_ , nenhuma andar solicitado

end

endcase

end

always begin //parte combinacional

case (aux\_s)

0: begin

HEX1 = 7'b1000000; //andar atual = 0

end

1: begin

HEX1 = 7'b1111001; //andar atual = 1

end

2: begin

HEX1 = 7'b0100100; //andar atual = 2

end

3: begin

HEX1 = 7'b0110000; //andar atual = 3

end

4: begin

HEX1 = 7'b0011001; //andar atual = 4

end

5: begin

HEX1 = 7'b0010010; //andar atual = 5

end

6: begin

HEX1 = 7'b0000010; //andar atual = 6

end

7: begin

HEX1 = 7'b1111000; //andar atual = 7

end

8: begin

HEX1 = 7'b0000000; //andar atual = 8

end

endcase

end

always @( CLOCK ) begin

if ( estado\_atual == parado ) begin

LED\_G = 1;

LED\_R = 0;

end

else begin

LED\_G = 0;

LED\_R = 1;

end

end

always @( posedge CLOCK2 ) begin //parte sequencial

rs = 0;

case (estado\_atual)

parado: begin

if (s > aux\_s ) begin

estado\_atual = subindo;

rs = 1;

end else if (s < aux\_s) begin

estado\_atual = descendo;

rs = 1;

end

end

subindo: begin

aux\_s = aux\_s + 1;

if (s == aux\_s) begin

estado\_atual = parado;

rs = 1;

end

end

descendo: begin

aux\_s = aux\_s - 1;

if (s == aux\_s ) begin

estado\_atual = parado;

rs = 1;

end

end

/\*inativo: begin

aux\_s = aux\_s - 1;

if (aux\_s == 0) begin

estado\_atual = parado;

end

end\*/

endcase

end

Reset\_Delay r0( .iCLK(CLOCK\_50),.oRESET(DLY\_RST));

LCD\_TEST u1(

// Host Side

.iCLK(CLOCK\_50),

.iRST\_N(DLY\_RST),

// LCD Side

.LCD\_DATA(LCD\_DATA),

.LCD\_RW(LCD\_RW),

.LCD\_EN(LCD\_EN),

.LCD\_RS(LCD\_RS),

.estado\_atual(estado\_atual),

.Reset(rs)

);

endmodule

**7.2. LCD\_TEST**

//------------------------------------//

// CIRCUITOS LÓGICOS 2 //

// AULA PRÁTICA: LCD //

// EXEMPLO 02 //

// //

// EUDISLEY ANJOS //

// CENTRO DE INFORMÃTICA //

// UNIVERSIDADE FEDERAL DA PARAÃBA //

//------------------------------------//

//Estados = Parado , subindo e descendo

Module LCD\_TEST ( //Host Side //Chamar lcd teste no main

iCLK,iRST\_N,

//LCD Side

LCD\_DATA,LCD\_RW,LCD\_EN,LCD\_RS, estado\_atual, Reset );

//Host Side

input iCLK,iRST\_N;

input wire [1:0] estado\_atual; //chamar na main

input Reset;

//LCD Side

output [7:0] LCD\_DATA;

output LCD\_RW,LCD\_EN,LCD\_RS;

// Internal Wires/Registers

reg [5:0] LUT\_INDEX;

reg [8:0] LUT\_DATA;

reg [5:0] mLCD\_ST;

reg [17:0] mDLY;

reg mLCD\_Start;

reg [7:0] mLCD\_DATA;

reg mLCD\_RS;

wire mLCD\_Done;

parameter LCD\_INTIAL = 0;

parameter LCD\_LINE1 = 5;

parameter LCD\_CH\_LINE = LCD\_LINE1+16;

parameter LCD\_LINE2 = LCD\_LINE1+16+1;

parameter LUT\_SIZE = LCD\_LINE1+32+1;

always@(posedge iCLK or negedge iRST\_N or posedge Reset)

begin

if(!iRST\_N || Reset)

begin

LUT\_INDEX <= 0;

mLCD\_ST <= 0;

mDLY <= 0;

mLCD\_Start <= 0;

mLCD\_DATA <= 0;

mLCD\_RS <= 0;

end

else

begin

if(LUT\_INDEX < LUT\_SIZE)

begin

case(mLCD\_ST)

0: begin

mLCD\_DATA <= LUT\_DATA[7:0];

mLCD\_RS <= LUT\_DATA[8];

mLCD\_Start <= 1;

mLCD\_ST <= 1;

end

1: begin

if(mLCD\_Done)

begin

mLCD\_Start <= 0;

mLCD\_ST <= 2;

end

end

2: begin

if(mDLY<18'h3FFFE)

mDLY <= mDLY+1;

else

begin

mDLY <= 0;

mLCD\_ST <= 3;

end

end

3: begin

LUT\_INDEX <= LUT\_INDEX+1;

mLCD\_ST <= 0;

end

endcase

end

end

end

/\*

-------------------------------------------------------------------

-- ASCII HEX TABLE

-- Hex Low Hex Digit

-- Value 0 1 2 3 4 5 6 7 8 9 A B C D E F

------\----------------------------------------------------------------

--H 2 | SP ! " # $ % & ' ( ) \* + , - . /

--i 3 | 0 1 2 3 4 5 6 7 8 9 : ; < = > ?

--g 4 | @ A B C D E F G H I J K L M N O

--h 5 | P Q R S T U V W X Y Z [ \ ] ^ \_

-- 6 | ` a b c d e f g h i j k l m n o

-- 7 | p q r s t u v w x y z { | } ~ DEL

-----------------------------------------------------------------------

-- Example "A" is row 4 column 1, so hex value is 8'h41"

-- \*see LCD Controller's Datasheet for other graphics characters available

\*/

always

begin

if (estado\_atual == 0) begin

case (LUT\_INDEX)

// Initial

LCD\_INTIAL+0: LUT\_DATA <= 9'h038;

LCD\_INTIAL+1: LUT\_DATA <= 9'h00C;

LCD\_INTIAL+2: LUT\_DATA <= 9'h001;

LCD\_INTIAL+3: LUT\_DATA <= 9'h006;

LCD\_INTIAL+4: LUT\_DATA <= 9'h080;

// Line 1

LCD\_LINE1+0: LUT\_DATA <= 9'h150; // P

LCD\_LINE1+1: LUT\_DATA <= 9'h161; // a

LCD\_LINE1+2: LUT\_DATA <= 9'h172; // r

LCD\_LINE1+3: LUT\_DATA <= 9'h161; // a

LCD\_LINE1+4: LUT\_DATA <= 9'h164; // d

LCD\_LINE1+5: LUT\_DATA <= 9'h16F; // o

LCD\_LINE1+6: LUT\_DATA <= 9'h120; //espaço

LCD\_LINE1+8: LUT\_DATA <= 9'h15F; // \_

//LCD\_LINE1+13: LUT\_DATA <= 9'h120; // SP

// Change Line

LCD\_CH\_LINE: LUT\_DATA <= 9'h0C0;

endcase

end

else if (estado\_atual == 1) begin

case (LUT\_INDEX)

// Initial

LCD\_INTIAL+0: LUT\_DATA <= 9'h038;

LCD\_INTIAL+1: LUT\_DATA <= 9'h00C;

LCD\_INTIAL+2: LUT\_DATA <= 9'h001;

LCD\_INTIAL+3: LUT\_DATA <= 9'h006;

LCD\_INTIAL+4: LUT\_DATA <= 9'h080;

// Line 1

LCD\_LINE1+0: LUT\_DATA <= 9'h153; // S

LCD\_LINE1+1: LUT\_DATA <= 9'h175; // u

LCD\_LINE1+2: LUT\_DATA <= 9'h162; // b

LCD\_LINE1+3: LUT\_DATA <= 9'h169; // i

LCD\_LINE1+4: LUT\_DATA <= 9'h16E; // n

LCD\_LINE1+5: LUT\_DATA <= 9'h164; // d

LCD\_LINE1+6: LUT\_DATA <= 9'h16F; // o

LCD\_LINE1+7: LUT\_DATA <= 9'h120; //espaço

LCD\_LINE1+8: LUT\_DATA <= 9'h12B; // +

LCD\_LINE1+9: LUT\_DATA <= 9'h120; //espaço

LCD\_LINE1+10: LUT\_DATA <= 9'h120; //espaço

LCD\_LINE1+11: LUT\_DATA <= 9'h120; //espaço

LCD\_LINE1+12: LUT\_DATA <= 9'h120; //espaço

LCD\_LINE1+13: LUT\_DATA <= 9'h120; //espaço

LCD\_LINE1+14: LUT\_DATA <= 9'h120; //espaço

LCD\_LINE1+15: LUT\_DATA <= 9'h120; //espaço

// Change Line

LCD\_CH\_LINE: LUT\_DATA <= 9'h0C0;

LCD\_LINE2+0: LUT\_DATA <= 9'h120; // S

LCD\_LINE2+1: LUT\_DATA <= 9'h120; // u

LCD\_LINE2+2: LUT\_DATA <= 9'h120; // b

LCD\_LINE2+3: LUT\_DATA <= 9'h120; // i

LCD\_LINE2+4: LUT\_DATA <= 9'h120; // n

LCD\_LINE2+5: LUT\_DATA <= 9'h120; // d

LCD\_LINE2+6: LUT\_DATA <= 9'h120; // o

LCD\_LINE2+7: LUT\_DATA <= 9'h120; //espaço

LCD\_LINE2+8: LUT\_DATA <= 9'h120; // +

LCD\_LINE2+9: LUT\_DATA <= 9'h120; //espaço

LCD\_LINE2+10: LUT\_DATA <= 9'h120; //espaço

LCD\_LINE2+11: LUT\_DATA <= 9'h120; //espaço

LCD\_LINE2+12: LUT\_DATA <= 9'h120; //espaço

LCD\_LINE2+13: LUT\_DATA <= 9'h120; //espaço

LCD\_LINE2+14: LUT\_DATA <= 9'h120; //espaço

LCD\_LINE2+15: LUT\_DATA <= 9'h120; //espaço

endcase

end

else if (estado\_atual == 2) begin

case (LUT\_INDEX)

// Initial

LCD\_INTIAL+0: LUT\_DATA <= 9'h038;

LCD\_INTIAL+1: LUT\_DATA <= 9'h00C;

LCD\_INTIAL+2: LUT\_DATA <= 9'h001;

LCD\_INTIAL+3: LUT\_DATA <= 9'h006;

LCD\_INTIAL+4: LUT\_DATA <= 9'h080;

LCD\_LINE1+0: LUT\_DATA <= 9'h120; // S

LCD\_LINE1+1: LUT\_DATA <= 9'h120; // u

LCD\_LINE1+2: LUT\_DATA <= 9'h120; // b

LCD\_LINE1+3: LUT\_DATA <= 9'h120; // i

LCD\_LINE1+4: LUT\_DATA <= 9'h120; // n

LCD\_LINE1+5: LUT\_DATA <= 9'h120; // d

LCD\_LINE1+6: LUT\_DATA <= 9'h120; // o

LCD\_LINE1+7: LUT\_DATA <= 9'h120; //espaço

LCD\_LINE1+8: LUT\_DATA <= 9'h120; // +

LCD\_LINE1+9: LUT\_DATA <= 9'h120; //espaço

LCD\_LINE1+10: LUT\_DATA <= 9'h120; //espaço

LCD\_LINE1+11: LUT\_DATA <= 9'h120; //espaço

LCD\_LINE1+12: LUT\_DATA <= 9'h120; //espaço

LCD\_LINE1+13: LUT\_DATA <= 9'h120; //espaço

LCD\_LINE1+14: LUT\_DATA <= 9'h120; //espaço

LCD\_LINE1+15: LUT\_DATA <= 9'h120; //espaço

// Change Line

LCD\_CH\_LINE: LUT\_DATA <= 9'h0C0;

// Line 2

LCD\_LINE2+0: LUT\_DATA <= 9'h144; // D

LCD\_LINE2+1: LUT\_DATA <= 9'h165; // e

LCD\_LINE2+2: LUT\_DATA <= 9'h173; // s

LCD\_LINE2+3: LUT\_DATA <= 9'h163; // c

LCD\_LINE2+4: LUT\_DATA <= 9'h165; // e

LCD\_LINE2+5: LUT\_DATA <= 9'h16E; // n

LCD\_LINE2+6: LUT\_DATA <= 9'h164; // d

LCD\_LINE2+7: LUT\_DATA <= 9'h16F; // o

LCD\_LINE2+8: LUT\_DATA <= 9'h120; //espaço

LCD\_LINE2+9: LUT\_DATA <= 9'h12D; // -

LCD\_LINE2+10: LUT\_DATA <= 9'h120; //espaço

LCD\_LINE2+11: LUT\_DATA <= 9'h120; //espaço

LCD\_LINE2+12: LUT\_DATA <= 9'h120; //espaço

LCD\_LINE2+13: LUT\_DATA <= 9'h120; //espaço

LCD\_LINE2+14: LUT\_DATA <= 9'h120; //espaço

LCD\_LINE2+15: LUT\_DATA <= 9'h120; //espaço

endcase

end

end

LCD\_Controller u0( //Host Side

.iDATA(mLCD\_DATA),

.iRS(mLCD\_RS),

.iStart(mLCD\_Start),

.oDone(mLCD\_Done),

.iCLK(iCLK),

.iRST\_N(iRST\_N),

// LCD Interface

.LCD\_DATA(LCD\_DATA),

.LCD\_RW(LCD\_RW),

.LCD\_EN(LCD\_EN),

.LCD\_RS(LCD\_RS),

.Reset(Reset) );

endmodule

**7.3. LCD**

//------------------------------------//

// CIRCUITOS LÃ“GICOS 2 //

// AULA PRÃTICA: LCD //

// EXEMPLO 02 //

// //

// EUDISLEY ANJOS //

// CENTRO DE INFORMÃTICA //

// UNIVERSIDADE FEDERAL DA PARAÃBA //

//------------------------------------//

module LCD ( input CLOCK\_50, // 50 MHz clock

input [3:0] KEY, // Pushbutton[3:0]

input [17:0] SW, // Toggle Switch[17:0]

output [6:0] HEX0,HEX1,HEX2,HEX3,HEX4,HEX5,HEX6,HEX7, // Seven Segment Digits

output [8:0] LEDG, // LED Green

output [17:0] LEDR, // LED Red

inout [35:0] GPIO\_0,GPIO\_1, // GPIO Connections

// LCD Module 16X2

output LCD\_ON, // LCD Power ON/OFF

output LCD\_BLON, // LCD Back Light ON/OFF

output LCD\_RW, // LCD Read/Write Select, 0 = Write, 1 = Read

output LCD\_EN, // LCD Enable

output LCD\_RS, // LCD Command/Data Select, 0 = Command, 1 = Data

inout [7:0] LCD\_DATA // LCD Data bus 8 bits

);

// All inout port turn to tri-state

assign GPIO\_0 = 36'hzzzzzzzzz;

assign GPIO\_1 = 36'hzzzzzzzzz;

// reset delay gives some time for peripherals to initialize

wire DLY\_RST;

Reset\_Delay r0( .iCLK(CLOCK\_50),.oRESET(DLY\_RST));

// Send switches to red leds

assign LEDR = SW;

assign LEDG = 9'h000;

// turn LCD ON

assign LCD\_ON = 1'b1;

assign LCD\_BLON = 1'b1;

LCD\_TEST u1(

// Host Side

.iCLK(CLOCK\_50),

.iRST\_N(DLY\_RST),

// LCD Side

.LCD\_DATA(LCD\_DATA),

.LCD\_RW(LCD\_RW),

.LCD\_EN(LCD\_EN),

.LCD\_RS(LCD\_RS)

);

// blank unused 7-segment digits

assign HEX0 = 7'b111\_1111;

assign HEX1 = 7'b111\_1111;

assign HEX2 = 7'b111\_1111;

assign HEX3 = 7'b111\_1111;

assign HEX4 = 7'b111\_1111;

assign HEX5 = 7'b111\_1111;

assign HEX6 = 7'b111\_1111;

assign HEX7 = 7'b111\_1111;

endmodule

**7.4. LCD\_Controller**

//------------------------------------//

// CIRCUITOS LÃ“GICOS 2 //

// AULA PRÃTICA: LCD //

// EXEMPLO 02 //

// //

// EUDISLEY ANJOS //

// CENTRO DE INFORMÃTICA //

// UNIVERSIDADE FEDERAL DA PARAÃBA //

//------------------------------------//

module LCD\_Controller (// Host Side

input [7:0] iDATA,

input iRS,

input iStart,

output reg oDone,

input iCLK,iRST\_N,

// LCD Interface

output [7:0] LCD\_DATA,

output LCD\_RW,

output reg LCD\_EN,

output LCD\_RS,

input Reset );

parameter CLK\_Divide = 16;

// Internal Register

reg [4:0] Cont;

reg [1:0] ST;

reg preStart,mStart;

/////////////////////////////////////////////

// Only write to LCD, bypass iRS to LCD\_RS

assign LCD\_DATA = iDATA;

assign LCD\_RW = 1'b0;

assign LCD\_RS = iRS;

/////////////////////////////////////////////

always@(posedge iCLK or negedge iRST\_N or posedge Reset)

begin

if(!iRST\_N || Reset)

begin

oDone <= 1'b0;

LCD\_EN <= 1'b0;

preStart<= 1'b0;

mStart <= 1'b0;

Cont <= 0;

ST <= 0;

end

else

begin

////// Input Start Detect ///////

preStart<= iStart;

if({preStart,iStart}==2'b01)

begin

mStart <= 1'b1;

oDone <= 1'b0;

end

//////////////////////////////////

if(mStart)

begin

case(ST)

0: ST <= 1; // Wait Setup

1: begin

LCD\_EN <= 1'b1;

ST <= 2;

end

2: begin

if(Cont<CLK\_Divide)

Cont <= Cont + 1'b1;

else

ST <= 3;

end

3: begin

LCD\_EN <= 1'b0;

mStart <= 1'b0;

oDone <= 1'b1;

Cont <= 0;

ST <= 0;

end

endcase

end

end

end

endmodule

7.5. Reset\_Delay

//------------------------------------//

// CIRCUITOS LÃ“GICOS 2 //

// AULA PRÃTICA: LCD //

// EXEMPLO 02 //

// //

// EUDISLEY ANJOS //

// CENTRO DE INFORMÃTICA //

// UNIVERSIDADE FEDERAL DA PARAÃBA //

//------------------------------------//

module Reset\_Delay( input iCLK, output reg oRESET);

reg [19:0] Cont;

always@(posedge iCLK)

begin

if (Cont!=20'hFFFFF) begin

Cont <= Cont + 1'b1;

oRESET <= 1'b0;

end

else

oRESET <= 1'b1;

end

endmodule